(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2001－320386
（P2001－320386A）
(43)公開日　平成13年11月16日(2001.11.16)

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ	テーマコード（参考）
Ｈ０４Ｌ　12/28	２００	Ｈ０４Ｌ　12/28	２００Ｚ

審査請求　有　　請求項の数７　ＯＬ　（全 20 頁）

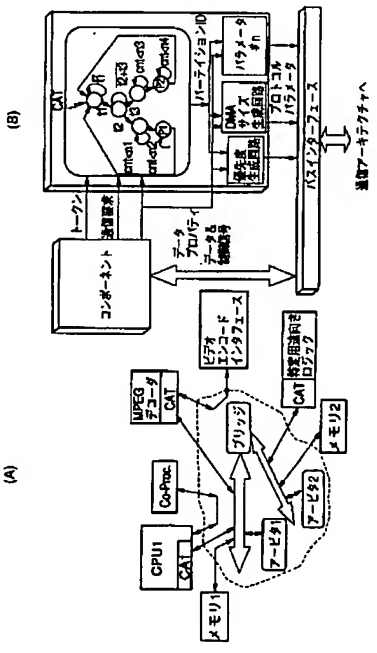
(21)出願番号　特願2001－58536(P2001－58536)
(22)出願日　平成13年３月２日(2001.3.2)
(31)優先権主張番号　０９／５７６９５５
(32)優先日　平成12年５月24日(2000.5.24)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　６０／１８６７５１
(32)優先日　平成12年３月３日(2000.3.3)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　アナンド・ラグナサン
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内
(74)代理人　100097157
弁理士　桂木　雄二

最終頁に続く

(54)【発明の名称】　電子システム

(57)【要約】
【課題】　フレキシブルで、システムコンポーネントの変動する通信要求に適応可能な、カスタムシステムオンチップ通信アーキテクチャの設計の一般的方法を提供する。
【解決手段】　複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムにおいて、少なくとも１つのコンポーネントは通信アーキテクチャチューナを有し、このチューナが、電子システムの変化する通信要求に電子システムが適応することを可能にする。基礎となる通信アーキテクチャトポロジーを、接続されるコンポーネントの変化する通信要求に適応可能にすることによって、最適化することができる。例えば、重要なデータを別個に処理することにより、通信レイテンシを小さくすることが可能である。この結果、サービス品質（ＱｏＳ）が大幅に改善される。



1

【特許請求の範囲】

【請求項１】　複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムにおいて、

少なくとも１つのコンポーネントは通信アーキテクチャチューナを有し、前記チューナは、電子システムの変化する通信要求に電子システムが適応することを可能にすることを特徴とする電子システム。

【請求項２】　複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムにおいて、

基礎となる通信アーキテクチャは少なくとも１つの通信アーキテクチャチューナを有する回路層を有し、前記チューナは、電子システムの変化する通信要求に電子システムが適応することを可能にすることを特徴とする電子システム。

【請求項３】　前記通信アーキテクチャチューナは、部分的にソフトウェアで実装されることを特徴とする請求項１記載の電子システム。

【請求項４】　前記通信アーキテクチャチューナは、複数のパーティションを検出するとともに、通信トランザクションによって満たされなければならない条件を検出して前記複数のパーティションのうちの１つの下にトランザクションを分類する少なくとも１つのパーティションディテクタと、

前記パーティションディテクタによって生成されるパーティションＩＤに基づいて、通信プロトコルパラメータの値を計算する少なくとも１つのパラメータ生成回路と、

をさらに有することを特徴とする請求項１記載の電子システム。

【請求項５】　前記通信アーキテクチャチューナは、複数のパーティションを検出するとともに、通信トランザクションによって満たされなければならない条件を検出して前記複数のパーティションのうちの１つの下にトランザクションを分類する少なくとも１つのパーティションディテクタと、

前記パーティションディテクタによって生成されるパーティションＩＤに基づいて、通信プロトコルパラメータの値を計算する少なくとも１つのパラメータ生成回路と、

をさらに有することを特徴とする請求項２記載の電子システム。

【請求項６】　前記通信アーキテクチャチューナは、部分的にソフトウェアで実装されることを特徴とする請求項２記載の電子システム。

【請求項７】　複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムの制御方法において、

複数のパーティションを検出してパーティション識別子

2

（ＩＤ）を生成し、

通信トランザクションによって満たされなければならない条件を検出し、

前記複数のパーティションのうちの１つの下にトランザクションを分類し、

前記パーティションＩＤに基づいて通信プロトコルパラメータの値を計算することで前記電子システムの変化する通信要求に電子システムが適応することを可能にすることを特徴とする制御方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】［Ｉ．Ａ．］本発明はカスタムシステムオンチップ通信アーキテクチャに係り、特に通信アーキテクチャを設計する方法および新規な電子システムに関する。

【０００２】

【従来の技術】［Ｉ．Ｂ．］電子システム設計におけるシステムオンチップ（ＳＯＣ）パラダイムの発展は、システムのコスト、サイズ、パフォーマンス、消費電力、および設計ターンアラウンドタイムの改善など、設計者にいくつかの利益を提供する可能性がある。この可能性を実現する能力は、設計者がシステムオンチップアプローチによって提供されるカスタム化可能性をどれくらいうまく活用するかに依存する。このカスタム化可能性の１つの側面は、システムを構成するために使用されるコンポーネント（例えば、プロセッサや、領域固有のコア、周辺機器など）の多様性および設定可能性(configurability)に現れるが、もう１つの、同じように重要な側面は、システム通信アーキテクチャのカスタム化可能性である。オンチップ通信に対するますます多様化する多くの要求をサポートするために、厳しいパフォーマンス制約およびパワーバジェットを満たしながら、通信アーキテクチャを、使用されるターゲットシステムあるいはアプリケーション領域にカスタム化することが必要とされる。

【０００３】Ｉ．Ｂ．１．関連する研究

本発明について従来技術との関係で説明するために、システムレベル設計、ハードウェア・ソフトウェア協調設計(HW/SW codesign)、およびネットワーキングプロトコルの分野における関連する研究についてここで説明する。あらかじめ設計されたコアおよびアプリケーション固有のハードウェアへの、アプリケーションタスクのＨＷ／ＳＷパーティショニングおよびマッピングを通じて、アプリケーション固有のアーキテクチャをシステムレベルで合成することに関連する多くの研究がある。詳細には、以下の文献参照。

・D. D. Gajski, F. Vahid, S. Narayan and J. Gong, "Specification and Design of Embedded Systems", Prentice Hall, 1994

・G. De Micheli, "Synthesis and Optimization Digit

al Circuits", McGraw-Hill, New York, NY, 1994
・R. Ernst, J. Henkel, and T. Benner, "Hardware-software cosynthesis for microcontollers", IEEE Design and Test Magazine, pp.64-75, Dec. 1993
・T. B. Ismali, M. Abid, and M. Jerraya, "COSMOS: A codesign approachfor a communicating system", in Proc. IEEE International Workshop on Software/Codesign, pp.17-24, 1994
・A. Kalavade and E. Lee, "A globally critical/locally phase driven algorithm for the constrained hardware software partitioning problem in Proc. IEEE International Workshop on Hardware/Software Codesign, pp.42-48,1994
・P. H. Chou, R. B. Ortega, and G. B. Borriello, "The CHINOOK hardware/software cosynthesis system", in Proc. Int. Symp. System Level Synthesis, pp.22-27, 1995
・B. Lin, "A system design methodology for software/hardware codevelopment of telecommunication network applications", in Proc. Design Automation Conf., pp.672-677, 1996
・B. P. Dave, G. Lakshminarayana, and N. K. Jha, "COSYN: hardware-software cosynthesis of embedded systems", in Proc. Design Automation Conf.,pp.703-708, 1997
・P. Knudsen and J. Madsen, "Integrating communication protocol selection with partitioning in hardware/software codesign", in Proc. Int. Symp. System Level Synthesis, pp.111-116, Dec. 1998

【０００４】これらの従来技術のうちの一部は、ＨＷ／ＳＷパーティショニングおよびマッピング中に通信の効果の影響を考慮しようとしているが、それらは、一定の通信プロトコル（例えば、ＰＣＩバス）を仮定するか、あるいは、いくつかのプロトコル選択肢からなる「通信ライブラリ」から選択を行うものである。通信アーキテクチャのシステムレベル合成に関する研究のほとんどは、通信アーキテクチャトポロジーの合成、すなわち、コンポーネントが専用リンクまたは共有通信チャネル（バス）を通じてどのように構造的に接続されるかを扱うものである。これらのアーキテクチャに関してさらに詳細には、以下の文献を参照のこと。

・T. Yen and W. Wolf, "Communication synthesis for distributed embedded systems", in Proc. Int. Conf. Computer-Aided Design, pp.288-294, Nov.1995
・J. Daveau, T. B. Ismail, and A. A. Jerraya, "Synthesis of system-level communication by an allocation based approach", in Proc. Int. Symp. System Level Synthesis, pp.150-155, Sept. 1995
・M. Gasteier and M. Glesner, "Bus-based communication synthesis on system level", in ACM Trans. Design Automation Electronic Systems, pp.1-11, Jan. 1999
・R. B. Ortega and G. Borriello, "Communication synthesis for distributed embedded systems", in Proc. Int. Conf. Computer-Aided Design, pp.437-444, 1998

【０００５】トポロジー選択は通信アーキテクチャ設計における重要なステップであるが、同様に重要なのは、選択されたトポロジーにおいてチャネル／バスによって使用されるプロトコルの設計である。例えば、システムコンポーネントによって生成される通信トラフィックの性質は、場合によっては、タイムスライス型バスプロトコルの使用に有利なこともあり、静的優先度プロトコルの使用に有利なこともある。さらに詳細には、以下の文献を参照のこと。

・"Sonics Integration Architecture, Sonics Inc. (http://www.sonicsinc.com/)"
・On-Chip Bus Development Working Group Specification I Version 1.1.0.VSI Alliance, Aug. 1998

ＶＳＩアライアンス(VSI Alliance)のオンチップバスワーキンググループ(on-chip bus working group)は、広範囲のＳＯＣ通信要求を満たすために多くのプロトコルが必要となることを認識している（On-Chip Bus Development Working Group Specification I Version 1.1.0. VSI Alliance, Aug. 1998、参照）。さらに、ほとんどのプロトコルは、アービトレーション優先度、転送ブロックサイズなどのようなパラメータの形で、カスタム化のための手段を設計者に提供する。これらのパラメータに適当な値を選ぶことは、コンポーネント間通信に伴うレイテンシおよび転送帯域幅に大きな影響を及ぼすことがある。

【０００６】最後に、コンポーネントとバスの間あるいはコンポーネントどうしの間のインタフェースの効率的なハードウェア実装を自動的に生成することを扱うインタフェース合成についての多くの研究がある。詳細には、以下の文献参照。

・G. Borriello and R. H. Katz, "Synthesis and optimization of interface transducer logic", in Proc. Int. Conf. Computer Design, Nov. 1987
・J. S. Sun and R. W. Brodersen, "Design of system interface modules",in Proc. Int. Conf. Computer-Aided Design, pp.478-481, Nov. 1992
・P. Gutberlet and W. Rosenstiel, "Specification of interface components for synchronous data paths", in Proc. Int. Symp. System Level Synthesis, pp.134-139, 1994
・S. Narayanan and D. D. Gajski, "Interfacing incompatible protocols using interface process generation", in Proc. Design Automation Conf., pp.468-473, June 1995

・P. Chou, R. B. Ortega, and G. Borriello, "Interface co-synthesis techniques for embedded systems", in Proc. Int. Conf. Computer-Aided Design, pp.280-287, Nov. 1995
・J. Oberg, A. Kumar, and A. Hemani, "Grammar-based hardware synthesisof data communication protocols", in Proc. Int. Symp. System Level Synthesis, pp.14-19, 1996
・R. Passerone, J. A. Rowson, and A. Sangiovanni-Vincentelli, "Automatic synthesis of interfaces between incompatible protocols", in Proc. Design Automation Conf., pp.8-13, June 1998
・J. Smith and G. De Micheli, "Automated composition of hardware components", in Proc. Design Automation Conf., pp.14-19, June 1998

これらの技術は、指定されたプロトコルの実装に関する問題を解決しているが、プロトコル自体のカスタム化には対処していない。

【０００７】まとめると、システムレベル設計およびＨＷ／ＳＷ協調設計の分野における従来技術は、ＳＯＣ通信アーキテクチャで用いられるプロトコルをアプリケーションの必要に合わせてカスタム化するという問題に十分に対処していない。さらに、従来の研究では、通信アーキテクチャの設計は、アプリケーションおよびその環境（例えば、代表的な入力トレース）に関する情報を用いて静的に実行されている。しかし、いくつかのアプリケーションでは、各コンポーネントによって要求される通信帯域幅、通信する必要のあるデータ量、および、各通信要求の相対的「重要性」は、大きな動的変動を受けることがある。このような状況では、従来の通信アーキテクチャで用いられるプロトコルは、基礎となる通信トポロジーを、アプリケーションの変動する要求を満たすように適応させることができないことがある。

【０００８】通信およびネットワーキングプロトコルの分野では、コネクション設定遅延や故障確率、スループット、残留誤り率などのような様々なサービス品質（ＱｏＳ）パラメータを満たすためのプロトコルの設計に多くの研究が向けられている。これらのパラメータについて詳細には、A. S. Tanenbaum, "Computer Networks", Englewood Cliffs, NJ, Prentice Hall, 1989、参照。上記のメトリック（パラメータ）を改善するようにプロトコルを適応させるための、フローおよびトラフィックの制御アルゴリズムのような高度な従来技術が提案されている。

【０００９】システムオンチップ通信アーキテクチャは、複雑さが増大するとともに、通信ネットワークの分野で開発された技術を利用することによって進化することが必要とされる。しかし、レイテンシ要求、エラー許容範囲および耐障害性(resilience)要求のように（これらには限定されないが）、ここで扱われる問題点と通信ネットワークプロトコル設計で遭遇する問題点とを区別する重要な相違点がある。

【００１０】Ｉ．Ｂ．２．通信アーキテクチャチューナ：概要と設計の問題点

このセクションでは、従来の通信アーキテクチャはフレキシビリティが制限され、システムコンポーネントの変動する通信要求に適応することができないために、システムのパフォーマンスが大幅に劣化する可能性があることを示すことによって、ＣＡＴベースの（ＣＡＴに基づく）通信アーキテクチャの必要性を実証する。

【００１１】例１：　図１に示すシステム例を考える。このシステムは、ネットワークインタフェースカードで用いられるＴＣＰ／ＩＰ通信プロトコルの一部を表す（以下、このシステムをＴＣＰシステムという）。図１のシステムは、チェックサムによる符号化（出力パケットに対して）および誤り検出（入力パケットに対して）を実行し、イーサネット（登録商標）コントローラ周辺機器（これは、物理層およびリンク層のネットワークプロトコルを実装する）とのインタフェースを行う。ＴＣＰプロトコルにおけるパケットはサービス品質（ＱｏＳ）の概念を含まないため、パケットデータ構造は、パケットが処理されるべきデッドラインを示すフィールドをヘッダに含むように拡張されている。システムの実行中の目的は、デッドライン超過のパケット数を最小にすることである。

【００１２】図１（ａ）に、並行通信タスクあるいはプロセスのセットとしてＴＣＰシステムの動作を示す。ネットワークからＴＣＰシステムにより受信されるパケットに対してＴＣＰシステムによって実行されるタスクについて説明する。プロセスether#driver（イーサネットデバイスドライバを表す）は、イーサネットコントローラ周辺機器からデータを読み出し、共有システムメモリ内にパケットを作成する。プロセスpkt#queueは、パケットヘッダからの選択された情報を含むキューを管理する。プロセスip#checkは、上記のキューからパケット情報を取り出し、パケットヘッダ内のいくつかの特定のフィールドをゼロで上書きし、チェックサム計算の準備をする。プロセスchecksumは、共有メモリからパケットを取り出し、各パケットのチェックサム値を計算し、その値をip#checkプロセスに返し、必要に応じてエラーのフラグを立てる。

【００１３】図１（ｂ）に、ＴＣＰシステムを実装するために用いられるシステムアーキテクチャを示す。ether#driverプロセスおよびpkt#queueプロセスは、ＭＩＰＳＲ３０００プロセッサ上で動作する組込みソフトウェアにマッピングされ、ip#checkプロセスおよびchecksumプロセスは、専用ハードウェアを用いて実装される。システムコンポーネント間のすべての通信は、共有バスを用いて実装される。共有バスで用いられるプロトコルは、静的優先度に基づくアービトレーションおよびＤＭ

Ａモード転送をサポートする。ここで、ＤＭＡモード転送という用語は、単一のバスワードより大きいクラスタあるいはチャンクでのデータの伝送を指すために用いられる。静的優先度に基づくアービトレーションでは、バスに接続される各コンポーネントには固定優先度が割り当てられる。いつでも、アービタは、要求するコンポーネントにバスの使用を最高優先度で許可する。これらのチャンクの粒度(granularity)は、各コンポーネントに割り当てられるＤＭＡサイズパラメータの値によって支配される。

【００１４】バスアービタおよびコンポーネントのバスインタフェースはともにバスプロトコルを実装する。バスプロトコルにより、システム設計者は、各コンポーネントのバス優先度およびＤＭＡブロックサイズのようなさまざまなパラメータの値を指定することができる。

【００１５】バスプロトコルパラメータのいくつかの異なる値に対する図１のＴＣＰシステムのパフォーマンスを解析する。この実験では、説明を簡単にするため、各コンポーネントのバス優先度値のみを考え、残りのプロトコルパラメータの値は固定する。デッドラインの余裕時間(laxity)をさまざまに変えたパケットのトレースを用いて、システムシミュレーションを実行した。４個のパケット（ｉ，ｉ＋１，ｊ，ｊ＋１と番号をつける）を処理するＴＣＰシステムの実行の概要図を図２に示す。この図は、各パケットがネットワークから到着する時刻と、いつまでに処理される必要があるかのデッドラインとを示す。注意すべき点として、パケットの到着時刻はｉ，ｉ＋１，ｊ，ｊ＋１の順序であるが、デッドラインは異なる順序ｉ＋１，ｉ，ｊ，ｊ＋１である。この説明のために、２つの異なるバス優先度割当て（checksum ＞ ip#check ＞ ether#driver、および、ether#driver ＞ip#check ＞ checksum）に注目する。他の優先度割当てについてはここでは明示的に考慮しないが、上記の２つの場合のいずれかについて提示する議論が他のあらゆる優先度割当てについても成り立つことは、当業者には明らかである。

【００１６】図２の第１のケースは、バス優先度割当て checksum ＞ ip#check ＞ ether#driver が用いられるときのシステムの実行を表す。パケットｉに対するether#driverプロセスの完了後、アービタは、次のように２つの競合するバスアクセス要求を受け取る。すなわち、プロセスip#checkがパケットｉを処理するためにバスアクセスを要求する一方、ether#driverがパケットｉ＋１を処理するためにバスアクセスを要求する（パケットｉ＋１はすでにネットワークから到着しているため）。用いられている優先度割当てに基づいて、アービタは、バスアクセス権をプロセスip#checkに与える。これは、実質的に、ip#checkおよびchecksumがパケットｉの処理を完了するまでパケットｉ＋１の処理を遅延させる。これにより、パケットｉ＋１がそのデッドラインに遅れることになる。パケットｊおよびｊ＋１は、図２に示されるように、それらのデッドラインに間に合う。一般に、デッドラインが到着時刻と同じ順序でないようなパケットの系列に対して、優先度割当て（checksum ＞ ip#check ＞ ether#driver）はデッドラインを超過する可能性がある。

【００１７】バスプロトコルに対して異なる優先度割当て（ether#driver ＞ ip#check ＞checksum）を用いることによって上記の問題点の解消を試みる。この新しい優先度割当ての下でのシステムの実行を、図２の第２のケースに示す。新しい優先度割当ての結果として、パケットｉ＋１が到着すると、プロセスether#driverは、パケットｉの完了を待たずにパケットｉ＋１を処理することができる。この結果、パケットｉおよびｉ＋１のいずれのデッドラインも間に合う。しかし、デッドラインが到着時刻と同じ順序のパケットｊおよびｊ＋１を考える。プロセスether#driverがパケットｊの処理を完了した後、共有バスに対する競合が、パケットｊ＋１に対するプロセスether#driverと、パケットｊに対するプロセスip#checkの間に生じる。選択された優先度割当てに基づいて、アービタは、プロセスether#driverを優先するように決定する。これは、パケットｊに対するプロセスip#checkおよびchecksumの実行を遅延させ、システムがパケットｊのデッドラインを超過することにつながる。

【００１８】まとめると、ＴＣＰシステムについて考察したこれらの２つのバス優先度割当てはそれぞれデッドラインを超過することにつながる。さらに、前に２つの段落で提示した議論を適用して、あらゆる可能な優先度割当てについて、パケットｉ＋１またはパケットｊがそのデッドラインを超過することになることを示すことができる。

【００１９】ＴＣＰの例においてデッドラインを超過することにつながる通信アーキテクチャの欠点は、以下のようにまとめることができる。さまざまなシステムコンポーネント（ether#driver、ip#check、およびchecksum）によって生成される通信トランザクションの相対的重要性は、それらが処理しているパケットのデッドラインに依存して変わる。一般に、各通信トランザクションの重要度あるいはクリティカル性(criticality)は、通信がシステムのクリティカルパスにあるかどうかをともに決定するいくつかのファクタに依存することがある。通信アーキテクチャは、重要な通信要求と重要でない通信要求を区別し、それに従ってそれらを処理することができる必要がある。

【００２０】

【発明が解決しようとする課題】ＴＣＰの例で示したように、従来の通信アーキテクチャは、少なくとも次のような課題を有する。

（ｉ）提供されるカスタム化可能性の程度は、厳しいパフォーマンス要求のシステムには不十分なことがある。

（ｉｉ）一般に、システムの変動する通信要求と、通信されるデータの変化する性質を検知してそれに適応することができない。

【００２１】

【課題を解決するための手段】［ＩＩ．本発明の技術の要約］本発明は、フレキシブルで、システムコンポーネントの変動する通信要求に適応可能な、カスタムシステムオンチップ通信アーキテクチャの設計の一般的方法を提供する。本発明の技術を用いて、基礎となる通信アーキテクチャトポロジーを、接続されるコンポーネントの変化する通信要求に適応可能にすることによって、最適化することができる。例えば、重要なデータを別個に処理することにより、通信レイテンシを小さくすることが可能である。この結果、システム全体のパフォーマンス、観測される通信帯域幅およびバス利用率、ならびに、重要なデッドラインを守るシステムの能力などの、さまざまなサービス品質（ＱｏＳ）が大幅に改善される。

【００２２】本発明の技術は、各コンポーネントに通信アーキテクチャチューナ（ＣＡＴ：Communication Architecture Tuner）という回路の層を付加することに基づいている。ＣＡＴは、システムコンポーネントの内部状態およびシステムコンポーネントによって生成される通信トランザクションの内部状態をモニタし解析して、さまざまなシステムレベルのパフォーマンスメトリックに対するそれらの影響に関して通信トランザクションの相対的重要性を「予測」する。この解析の結果は、コンポーネントの変化する通信要求に最もよく適合するように、基礎となる通信アーキテクチャのパラメータを設定するために、ＣＡＴによって使用される。

【００２３】本発明の目的を達成するために、複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムにおいて、少なくとも１つのコンポーネントは、通信アーキテクチャチューナを有し、前記チューナは、電子システムが、電子システムの変化する通信要求に適応することを可能にする。

【００２４】本発明のもう１つの側面によれば、複数の共有通信チャネルによって相互接続された複数のコンポーネントを有する電子システムにおいて、基礎となる通信アーキテクチャは、少なくとも１つの通信アーキテクチャチューナを有する回路層を有し、前記チューナは、電子システムが、電子システムの変化する通信要求に適応することを可能にする。

【００２５】好ましくは、通信アーキテクチャチューナは、部分的にソフトウェアで実装される。

【００２６】好ましくは、通信アーキテクチャチューナは、さらに、あるパーティションの下にトランザクションを分類するために複数のパーティションと通信トランザクションによって満たされなければならない条件とを検出する少なくとも１つのパーティションディテクタと、パーティションディテクタによって生成されるパーティションＩＤに基づいて通信プロトコルパラメータに対する値を計算する少なくとも１つのパラメータ生成回路とを有する。

【００２７】

【発明の実施の形態】［ＩＶ．Ａ．説明のロードマップ］プロトコルの静的カスタム化がシステムの時間変化する通信要求を完全に満たすことはできないようなシステム例および状況を解析することによって、ＣＡＴ（通信アーキテクチャチューナ）ベースの通信アーキテクチャの必要性を説明する。続いて、ＣＡＴベースの通信アーキテクチャの設計に関連する問題点およびトレードオフについて説明する。次に、パフォーマンス改善の可能性を最大限に活用するためには、ＣＡＴのハードウェア実装の複雑さを考慮する必要があることを示す。次に、ＣＡＴベースのＳＯＣ通信アーキテクチャの設計に対する一般的な方法およびアルゴリズムを提示する。通信アーキテクチャトポロジー、代表的な入力トレース、およびターゲットパフォーマンスメトリックが規定されたシステムに対して、本発明の技術は、そのシステムにおけるさまざまなチャネル／バスに対する最適化された通信プロトコルを決定するために使用される。その後、各コンポーネントと通信アーキテクチャとの間に接続されるＣＡＴの形で、効率的なハードウェア実装について説明する。本発明の技術は、さらに、ＡＴＭスイッチポートスケジューラおよびＴＣＰ／ＩＰネットワークインタフェースカードサブシステムを含む、いくつかのシステム例について、実験結果とともに説明される。テスト結果によれば、ＣＡＴベースの通信アーキテクチャを有するシステムについてのパフォーマンスメトリック（例えば、デッドライン超過の数、平均または総計処理時間など）は、従来の最適化された通信アーキテクチャを有するシステムよりも大幅に（ときには、１桁以上）良好であることが示される。

【００２８】要約すれば、以下の通りである。

【００２９】・ＣＡＴベースの通信アーキテクチャは、基礎となる通信アーキテクチャの能力を拡張することができる。ＣＡＴベースの通信アーキテクチャが、それに接続された各コンポーネントに対して提示する（通信レイテンシや帯域幅のような）タイミング挙動は、コンポーネントの要求に応じてよりよくカスタム化され、変えられる。この結果、システムパフォーマンスが大幅に改善される。

【００３０】・本発明によるＣＡＴ設計方法は、通信アーキテクチャプロトコルの高度化と、付加されるハードウェアの複雑さ（したがって、それにより生じるオーバーヘッド）とのバランスをとる。

【００３１】・いくつかの場合には、ＣＡＴベースの通信アーキテクチャを用いると、プロトコルパラメータの静的カスタム化に基づくものよりもシステムの性能を大

幅に改善することができる。

【００３２】［ＩＶ．Ｂ．ＣＡＴベースの通信アーキテクチャ］このサブセクションでは、ＣＡＴベースの通信アーキテクチャを示し、このようなアーキテクチャが上記の欠点をどのように解決するかについて説明する。その後、ＣＡＴベースの通信アーキテクチャ設計に伴う主な問題点および妥協点（トレードオフ）について議論する。

【００３３】ＣＡＴベースの通信アーキテクチャは、それに接続されたさまざまなコンポーネントの変化する要求に従って、基礎となる通信アーキテクチャを適応させる、ハードウェア層を用いることにより、従来の技術のセクションに記載した問題点を解決する。従来の技術のセクションに記載した例を用いて、ＣＡＴベースの通信アーキテクチャを利用してどのようにしてパフォーマンスを改善することができるかを示す。

【００３４】例２：　例１に関連して説明したＴＣＰシステムに対するＣＡＴベースの通信アーキテクチャを図３（ａ）に示す。注意すべき点であるが、この例は単なる例示であり、特許請求の範囲を限定することを意図するものではない。ＣＡＴが、ether#driver、ip#check、およびchecksumの各プロセスを実装するコンポーネントに付加される。さらに、バス制御ロジック（アービタおよびコンポーネントバスインタフェース）は、ＣＡＴの動作を容易にするように拡張される。ＣＡＴを有するコンポーネントの詳細図を図３（ｂ）に示す。コンポーネントは、通信要求を発生するときＣＡＴに通知する。また、ＣＡＴは、通信されるデータと、コンポーネントの内部状態とに関する選択された項目を観測する。

【００３５】この例では、ＣＡＴは、コンポーネントによって現在処理されているパケットのヘッダからの、パケットサイズフィールドおよびデッドラインフィールドを観測する。ＣＡＴは、以下の作用を実行する。

（ｉ）現在処理されているパケットのサイズおよびデッドラインに基づいて通信イベントをグループ分けする。

（ｉｉ）各グループのイベントに対して、さまざまなプロトコルパラメータへの値の適当な割当てを決定する。

その結果、通信アーキテクチャの特性（通信を実行するのに必要な時間を含む）は、通信要求の相異なる必要性と相対的重要度に従って適応される。デッドラインを用いる理由は、デッドラインの近いパケットには高い重要度を与える必要があるからである。パケットのサイズを用いる理由はさらに複雑である。システム内のすべてのパケットがほぼ同じデッドラインを有する場合、小さいパケットの完了を優先するほうが好ましい。その場合、それらのパケットのほうが、デッドラインに間に合う可能性が高いからである。

【００３６】本明細書で以下で説明する技術は、図３のＣＡＴベースのＴＣＰシステムアーキテクチャを実装するために使用される。説明を簡単にするため、ＣＡＴは、バス優先度のみを変化させるように使用した。他のすべてのパラメータは、図１のアーキテクチャで用いたものと同じ値に指定した。ＣＡＴは、コンポーネントから生成される通信要求を、それらの通信要求が属するパケットに基づいてグループ分けし、１つのパケットに関連するすべての通信要求の優先度を、式$s \times (t_d - t_a)$を用いて計算する。ただし、ｓ、t_dおよびt_aはそれぞれ、パケットサイズ、デッドライン、および到着時刻である。

【００３７】最適化されたシステムの実行を図４に示す。図２で従来の通信アーキテクチャの欠点を例示するために用いたのと同じパケット系列を、このアーキテクチャでも利用した。システムは、すべてのパケットについてデッドラインを満たす（図１に示したもとのシステムアーキテクチャでは、どの優先度割当てでもデッドラインを超過していたことを想起すべきである）。パケットｉ＋１（デッドラインが近い）が到着すると、ＣＡＴは、ether#driverによって生成される通信要求に、依然としてパケットｉを処理しているip#checkおよびchecksumからの要求に割り当てる優先度よりも高い優先度を割り当てる。しかし、パケットｊ＋１が到着すると、ether#driverによって生成される通信要求には、より低い優先度が割り当てられることにより、ip#checkおよびchecksumは、パケットｊの近いデッドラインを満たすためにパケットｊの処理を完了することが可能となる。

【００３８】従来の例１および本発明の例２から、ＣＡＴ（通信アーキテクチャチューナ）の必要性は明らかである。このチューナは、接続されるコンポーネントの変化する通信要求を検出し、システムが、必要なときに、その通信リソースを有効に活用することを可能にする。ＣＡＴベースの通信アーキテクチャの効果的な実現は、以下のステップを正しく実行することによりなされる。

・システムのパフォーマンス解析から、パフォーマンスクリティカル通信イベントを識別するステップ。

・システムの実行中に、ハードウェアにおけるこれらの通信イベントの発生を検出するステップ。

・（優先度やＤＭＡサイズのような）通信プロトコルパラメータに対する適当な値をクリティカルイベントに割り当てることにより、高いパフォーマンスの実装を実現するステップ。

【００３９】システムレベルのパフォーマンス解析のためのいくつかの技術がすでに提案されており、第１ステップで利用可能であるが、本明細書では、クリティカル通信イベントを識別するための基礎として、システム実行トレースの解析を利用する。システム実行トレースに関する背景的知識について詳細には、以下の文献参照。

・D. D. Gajski, F. Vahid, S. Narayan and J. Gong, "Specification and Design of Embedded Systems", Prentice Hall, 1994

・G. De Micheli, "Synthesis and Optimization Digit

al Circuits", McGraw-Hill, New York, NY, 1994

システムシミュレーションを通じて生成される実行トレースを利用することの重要な利点は、システムレベルのシミュレーションモデルが存在するような任意のシステムについて実行トレースを導出することができることである。生成されたトレースを解析して、個々の通信イベント（または通信イベントのグループ）がシステムのパフォーマンスに及ぼす影響を調べることができる。システムの「クリティカルパス」にある通信イベントであって、その遅延が、指定されたパフォーマンスメトリックに重大な影響を及ぼす通信イベントは、クリティカルであるとして分類することができる。クリティカル通信イベントを識別するために使用される技術の詳細は、セクションＩＶ．Ｃで説明する。

【００４０】システム実行トレースは、使用される入力トレースあるいは刺激に固有であるため、シミュレーショントレースにおけるクリティカル通信イベントを、システムの実行中に発生するクリティカル通信イベント（おそらくは異なる刺激の下での）と関連づける簡単な方法はない。例えば、システム実行の開始後の２０番目、２１番目、および２２番目のデータ転送が、システムパフォーマンスに強い影響を及ぼすことが示される通信トレースを考える。仮にこれらのデータ転送を早めれば、与えられた入力トレースに対するシステムパフォーマンスが大幅に改善される。通信プロトコルを改善するために、これらの考察を利用しなければならないと仮定する。明らかに、２０番目、２１番目、および２２番目のデータ転送が高い優先度を有するような単純なシステムでは、パフォーマンス利得は何ら実現されないであろう。システムの実行中に発生するイベントの系列は、トレースのものとは相当に異なり得るからである。クリティカル通信イベントを識別することに加えて、それらの発生を、その他の、システムの状態およびそれが処理しているデータの容易に検出可能な性質と関連づける必要がある。

【００４１】例えば、シミュレーショントレースの解析により、クリティカルなデータ転送の発生が、その転送を実行しているコンポーネントの動作中に遭遇する特定の分岐と高い相関を有することが明らかになった場合、その分岐の発生を、そのコンポーネントによって生成されるデータ転送のクリティカルさのプレディクタ（予測子、predictor）として用いることが可能である。次の例で、これらのプレディクタを設計する際のいくつかのトレードオフを調べる。

【００４２】例３：　図５に示すシステムを考える。このシステムは、通信ネットワークに送信する前にセキュリティの目的でデータを暗号化するために用いられる。コンポーネント１は、データを処理し、使用する符号化・暗号化方式を決定し、そのデータをコンポーネント２に送る。コンポーネント２は、データの符号化・暗号化をした後、共有バスを通じてそのデータを、ネットワークへのデータ送信を行う周辺機器に送る。図６に、システムバス上で生じるデータ転送を示す。y_i（ｉ＝１，...，ｎ）で示す影付きの楕円は、コンポーネント２からネットワーク周辺機器へのデータ転送を表す。コンポーネント２は固定レートでデータを転送しなければならず、各データ転送はデッドライン（図６では破線で示す）の前に行われなければならないと仮定する。システムの主要なパフォーマンスメトリックは、コンポーネント２によって時間に間に合って完了されるデータ転送の数である。通信トレースは、デッドラインが頻繁に満たされないことを示す。また、システム実行トレースの解析は、デッドラインを満たさなかった通信イベント（例えば、ｙ１およびｙ２）を識別する。さらに、この解析は、クリティカル通信イベント、すなわち、早めればシステムパフォーマンスを改善する可能性のある通信イベントも識別する。ｙｉは、ｘｉの後にのみ起こり得るため、ｘｉを早めることは、システムパフォーマンスを改善する方法の１つである。ｙｉがデッドラインを満たさないようなすべてのｘｉの集合をＳとする。システムのパフォーマンスは、Ｓに属するイベントの通信回数が改善される場合に、改善することができる。

【００４３】クリティカル通信イベントをシミュレーショントレースから分離した後、システムの実行中にこれらの要素を識別する方式を開発する必要がある。前述のように、これは、クリティカル通信イベントの発生を、システムの状態およびそれが処理しているデータに関する情報と関連づけることによってなされる。この例では、クリティカル通信イベントは、それを発生したコンポーネントの制御フロー履歴と関連づけられると仮定する。制御フローイベントは、コンポーネントが特定の動作を実行するときに１の値をとるブール変数として定義される。例えば、図５のコンポーネント１の動作は、制御フローイベントｅ１、ｅ２、ｅ３、およびｅ４で表される。一般に、$e_1, e_2, \ldots, e_n$が、通信要求がクリティカルであるかどうかを決定するために用いられる制御フローイベントである場合、この通信イベントの集合がクリティカルであるかどうかを分類するブール関数$f_{critical} = f(e_1, e_2, \ldots, e_n)$が定義される。

【００４４】この分類に用いられる制御フロー変数の数は、通信イベントの分類に大きい影響を及ぼす。よい分類は、一対一写像の性質（すなわち、クリティカルとして分類されたあらゆるイベントが実際にクリティカルであり、あらゆるクリティカルイベントがこの分類によって検出される）を有するべきである。この例において、ただ１つの変数のみを分類に利用することができると仮定する。ｅ３がクラシファイアとして選択されるとする。デッドラインを超過するすべての場合において、イベントｅ３が生じる。この考察に基づいて、$f_{critical}$＝ｅ３と選ぶことが可能である。しかし、ｅ３は、非ク

リティカル通信イベントとともに生じることも多い。ｅ３をクラシファイアとして用いた場合、クリティカルであると分類された通信イベントのうちの１６％しか実際にはクリティカルではない。したがって、ｅ３は通信イベントを誤って分類することがあり、優先度を誤って増大させて、システムパフォーマンスに悪影響を及ぼすことがある。

【００４５】図７（ａ）および（ｂ）はそれぞれ、$f_{critical}$ におけるクリティカル通信イベントの割合と、Ｓのうち $f_{critical}$ によってカバーされる割合を、分類を実行する変数の数に対してプロットした図である。ｘ軸は、分類を実行するために用いた変数の数を示す。例えば、２個の変数を使用する最良のクラシファイアは、クリティカル通信イベントの１００％を捕捉するが、それによって「クリティカル」であるとして分類される通信イベントのうち５０％しか実際にはクリティカルではない。注意すべき点であるが、この例において、変数の数が増大すると、$f_{critical}$ におけるクリティカル通信イベントの割合も増大する。その理由は、変数の数が増大すると、分類判定基準はより厳格になり、非クリティカルイベントはそのテストを通りにくくなるためである。しかし、同時に、図７（ｂ）に示されるように、クリティカルイベントを逸する可能性がある（使用される変数の数が増大すると、カバーされるＳの割合が減少していることに注意）。したがって、システムパフォーマンスを最大限に改善するためには、適当な変数の数と、適当な分類関数を正しく選択する必要がある。この例では、最適な結果は、３個の変数（ｅ１、ｅ２、およびｅ３）と、分類関数 $f_{critical}$ ＝ｅ１．（－ｅ２）．ｅ３とを用いることによって得られる（論理反転を、変数の上にバーをつける代わりに、－符号で表す）。これは、ほとんどのクリティカルイベントを識別し、非クリティカルイベントを識別することはほとんどない。

【００４６】［ＩＶ．Ｃ．通信アーキテクチャチューナの設計のための方法およびアルゴリズム］このセクションでは、ＣＡＴベースの通信アーキテクチャの設計のための、構造化された方法および自動化アルゴリズムについて説明する。セクションＩＶ．Ｃ．１では、全体の方法論について説明し、関連するさまざまなステップについて概説する。セクションＩＶ．Ｃ．２では、重要なステップを実行するために用いられるアルゴリズムを詳細に提示する。

【００４７】ＩＶ．Ｃ．１．アルゴリズムと方法論：概説

このセクションでは、設計フローの点から、本発明の技術について説明する。まず、システムを、複数のあらかじめ設計されたコアおよびアプリケーション固有のロジックに分割しマッピングする。システムの通信および接続性の要求に基づいて、通信アーキテクチャトポロジーを選択する。その後、本発明の技術を用いて、選択されたトポロジーを最適化することができる。本発明のアルゴリズムは、入力として、シミュレート可能な分割／マッピングされたシステム記述と、選択された通信アーキテクチャトポロジーと、代表的な環境刺激すなわち入力トレースと、パフォーマンスメトリックに関する目標・制約とをとる。パフォーマンスメトリックは、特定量の作業を完了するのにかかる時間（例えば、処理時間の重みつきまたは一様平均）に関して、あるいは、リアルタイム制約のあるアプリケーションに対して満たされる（または満たされない）出力デッドラインの数に関して、指定することが可能である。アルゴリズムの出力は、ターゲットシステムの最適化された通信プロトコルのセットである。ハードウェアの観点から、システムは、必要な場合には通信アーキテクチャチューナ（ＣＡＴ）の付加により、および、通信アーキテクチャにおけるさまざまなチャネルに対するコントローラ／アービタの修正により、改善される。

【００４８】本発明の技術を用いて生成されるＣＡＴベースの通信アーキテクチャを有する代表的なシステムを図８（ａ）に示す。システムは、プロセッサコア、メモリ、および周辺機器を含むいくつかのコンポーネントを有する。選択された通信アーキテクチャトポロジーは、破線の境界で囲まれている。選択されたトポロジーは、コンポーネント間（例えば、プロセッサとコプロセッサの間）の専用チャネルと、ブリッジにより接続された２つの共有バスとからなる。本発明の技術の結果として付加あるいは修正されたシステムの部分は、図８（ａ）では影付きで示される。本発明の技術は、専用および共有チャネルの任意の相互接続ネットワークとして表現することが可能な一般的な通信アーキテクチャトポロジーに適用可能である。

【００４９】ＣＡＴの１つのコンポーネントの詳細図を図８（ｂ）に示す。ＣＡＴは、「パーティションディテクタ」回路（図中では有限状態オートマトンとして示す）と、システム実行中にさまざまな通信アーキテクチャプロトコルパラメータの値を生成するパラメータ生成回路とからなる。次に、これらの回路の役割について簡単に説明する。

【００５０】パーティションディテクタ：　通信パーティションは、システム実行中にコンポーネントによって生成される通信トランザクションのサブセットとして記述される。各コンポーネントごとに、本発明のアルゴリズムは、いくつかのパーティションと、各パーティションの下に分類されるために通信トランザクションが満たさなければならない条件とを識別する。これらの条件は、パーティションディテクタ回路に組み込まれる。パーティションディテクタ回路は、コンポーネントによって生成される以下の情報をモニタし解析する。

【００５１】・コンポーネントが特定の動作を実行していることを示すために、コンポーネントによって生成さ

れるｔｒａｃｅｒ（トレーサ）トークン。コンポーネントは、純粋にＣＡＴの目的のためのこれらのトークンを生成するように拡張される。

【００５２】・コンポーネントによって生成される通信トランザクション開始要求。

【００５３】・コンポーネントによって生成される通信データの、その他のアプリケーション固有のプロパティ（例えば、データの相対的重要性を示す、データ内のフィールド）。

【００５４】パーティションディテクタは、トレーサトークンおよび通信要求の特定の系列を用いて、１つのパーティションに属する連続する通信トランザクションの開始と終了を識別する。例えば、正規表現ｔ１．ｘ＊．C^1およびｔ１．ｘ＊．C^5を用いて、パーティションCP_iに属する通信イベントを区切ることが可能である。これは、トークンｔ１に続いて生成された４番目から７番目までの通信トランザクションがパーティションCP_iに分類されることを意味する。

【００５５】セクションＩＶ．Ｃ．２では、各パーティションに対する開始および終了条件を自動的に計算する一般的技術について説明する。

【００５６】パラメータ生成回路：　この回路は、パーティションディテクタ回路によって生成されるパーティションＩＤと、システム設計者によって指定されるその他のアプリケーション固有のデータプロパティとに基づいて、通信プロトコルパラメータ（例えば、優先度、ＤＭＡ／ブロックサイズなど）の値を計算する。これらのパラメータの値は、通信アーキテクチャ内のアービタおよびコントローラに送られることにより、通信アーキテクチャの特性が変化する。パラメータ生成回路を設計する自動技術については、セクションＩＶ．Ｃ．２で説明する。

【００５７】ＣＡＴベースの通信アーキテクチャの作用について、図９の記号的波形を用いて説明する。最初の２個の波形は、コンポーネントによって生成されるトレーサトークンを表す。次の２個の波形はそれぞれ、通信トランザクションと、パーティションディテクタ回路の状態とを表す。パーティションディテクタ回路の状態は、まずＳ０からＳ１に変わり、その後、コンポーネントによって生成されたトレーサトークンに反応して、Ｓ１からＳ２に変わる。パーティションディテクタが状態Ｓ２に達した後、コンポーネントによって生成された４番目の通信トランザクションにより、パーティションディテクタは状態Ｓ３に遷移する。パーティションディテクタのＦＳＭが状態Ｓ３にあるときに生じるすべての通信トランザクションは、パーティションCP_iに属するとして分類される。５番目の波形は、優先度生成回路の出力を示す。優先度生成回路は、パーティションCP_iに属するすべての通信トランザクションに優先度レベル４を割り当てる。この優先度の増大により、図９の最後の波形に示すように、パーティションCP_iに属する通信トランザクションに伴う遅延が減少する。

【００５８】ＣＡＴベースの通信アーキテクチャを設計する全体的なアルゴリズムの実施例を図１０に示す。ステップ１で、後のステップで用いられる情報および統計量を導出するために、パーティショニング／マッピングされたシステム記述に対してパフォーマンス解析を実行する。この作業では、Lahiri et al.に記載されたパフォーマンス解析技術を使用する。これは、精度に関してはシステムシミュレーションを実行するのと同等である一方、反復法を用いるよりもずっと効率がよい。さらに詳細には、K. Lahiri, A. Raghunathan and S. Dey, "Fast Performance Analysis of Bus Based System-on-Chip Communication Architectures", inProc. Int. Conf. Computer-Aided Design, Nov. 1999、参照。この解析の出力は、通信解析グラフ（ＣＡＧ：communication analysis graph）である。これは、与えられた入力トレースの下でのシステムの実行の非常にコンパクトな表現である。グラフの頂点は、システム実行中にさまざまなコンポーネントによって実行された計算および抽象通信のクラスタを表す。グラフの辺は、さまざまな計算と通信の間の相互依存関係を表す。注意すべき点であるが、通信解析グラフは実質的に時間的に展開されるため、巡回的であり、システム仕様からの単一の計算オペレーションまたは通信のいくつかの相異なるインスタンスを含むことがある。通信解析グラフは、詳細なシステム実行トレースから必要十分な情報を抽出することによって構築される（K. Lahiri, A. Raghunathan and S. Dey, "Fast Performance Analysis of Bus Based System-on-Chip Communication Architectures", in Proc. Int. Conf. Computer-Aided Design. Nov. 1999、参照）。ＣＡＧは、システムクリティカルパス、平均処理時間、デッドライン超過数などのようなさまざまなパフォーマンス統計量を決定するために、容易に解析することができる。

【００５９】ステップ２で、通信頂点を、通信解析グラフにおいて、いくつかのパーティションにグループ分けする。このパーティショニングの主要な理由は、各パーティションが相異なる通信要求を有する可能性があり、したがって、システムパフォーマンスを最適化するために、通信プロトコルのパラメータ（例えば、優先度、ＤＭＡサイズなど）に割り当てられる値の、異なるセットを必要とする可能性があるからである。注意すべき点であるが、極端な場合、通信解析グラフにおける各通信頂点が相異なるパーティションに割り当てられる可能性もある。しかし、これは次の２つの欠点を有する。

（ｉ）ＣＡＴにより生じる領域および遅延のオーバーヘッドが膨大になることがある。

（ｉｉ）非常に小さいパーティションを用いると、ＣＡＴハードウェアが、入力トレースの変動に非常に敏感に

なる可能性がある。
本発明は、感度(sensitivity)という新規なメトリックを提案する。これは、セクションＩＶ．Ｃ．２．ａで、通信インスタンス（頂点）をパーティションにグループ分けするために用いられる。設計者がパーティションの最適な粒度(granularity)を選択することを可能にする技術も提示される。
【００６０】ステップ３は、各通信パーティションのさまざまな統計量を推定し、それに基づいて、ステップ４は、各パーティションについて、通信アーキテクチャパラメータ値の割当てを決定する。これらのステップの詳細については、セクションＩＶ．Ｃ．２．ｂで説明する。ステップ４の出力は、システム通信アーキテクチャの候補プロトコルのセットである。
【００６１】ステップ５は、ステップ４で導出した最適化プロトコルのシステムパフォーマンスを再評価する。パフォーマンス改善がある場合、パフォーマンス改善が得られなくなるまでステップ１～５を繰り返す。
【００６２】ステップ６は、ステップ４で決定した最適化プロトコルを実装するハードウェア（ＣＡＴ）の合成を扱う。ＣＡＴベースの通信アーキテクチャの可能性を十分に活用するために、ハードウェア実装の複雑さおよびオーバーヘッドを考慮することが重要である。セクションＩＶ．Ｃ．２．ｃで、パーティションディテクタおよびパラメータ生成回路を生成する問題を、データ点のセットに当てはめる最小複雑さ関数を生成する問題として定式化し、回帰理論からの周知の技術を用いてこれをどのようにして効率的に解くことができるかについて概説する（C. A.F. Seber, C. J. Wild, Non-linear Regression, Wiley, New York, 1989、参照）。
【００６３】ＩＶ．Ｃ．２．アルゴリズムと方法論：詳細
このセクションでは、上で概説したステップについてさらに詳細に説明する。通信イベントインスタンスのパーティションを得る技術を説明する。また、プロトコルパラメータ値の最適セットを選択する方法、および、通信イベントインスタンスをパーティションに分類するためのＣＡＴハードウェアを合成する方法について説明する。
【００６４】ａ）通信イベントインスタンスのプロファイリングおよびパーティショニング
このセクションでは、本発明の方法のパーティショニングステップ（図１０のステップ２）について詳細に説明する。パーティショニングステップの目的は、通信プロトコルによって一様に扱うことが可能な通信イベントインスタンスのセットを識別し単一のパーティションにクラスタ化することである。例えば、プロトコルは、与えられた１つのパーティションのすべての要素を、共有バスにアクセスするための同じ優先度を有するように定義することが可能である。
【００６５】本発明のアルゴリズムのステップ１によって生成される通信解析グラフは、システムのパフォーマンスを、その通信イベントの遅延の関数として測るのに十分な情報を含む。ステップ２で、ＣＡＧの解析を実行して、システムパフォーマンスに対する個々の通信インスタンス遅延の影響を測る。システムパフォーマンスに類似の影響を及ぼすインスタンスは、同じパーティションにまとめられる。インスタンスのパフォーマンス影響は、インスタンスの通信遅延が変化したときのシステムパフォーマンスにおける変化を捕捉する感度と呼ばれるパラメータによって測られる。以下の例で、パーティショニング手続きについて説明する。
【００６６】図１１に、システム例の代表的実行から生成されるＣＡＧの一部を示す。影付きの頂点c_1～c_4は、通信イベントのインスタンスを表す。頂点z_1およびz_2は、システムの最終出力を表す。最小化すべき目的関数は、量$t(z_1)+t(z_2)$である。ただし、$t(v)$は、ＣＡＧ内の頂点ｖの終了時刻である。
【００６７】通信インスタンスc_iに対するシステムパフォーマンスの感度を測るため、c_iの既存の遅延を値Δだけ摂動し、ＣＡＧにおけるc_iの推移ファンアウトを用いて、影響される頂点の開始時刻および終了時刻を再計算する。頂点の更新された終了時刻を用いて、システムパフォーマンスメトリックの変化を計算する。この例では、c_1の遅延を１０単位だけ摂動すると、z_1およびz_2は両方ともそれぞれ１０単位だけ遅延するが、c_2の遅延はz_1のみを遅延させる。同様に、c_3を遅延させると、z_2の終了時刻が１０単位だけ遅延する。c_1はクリティカルパス上にないため、これを摂動しても、システムパフォーマンスに及ぼす影響はない。
【００６８】上記の手続きを用いて、各通信インスタンスc_iに対する感度$s(c_i)$を計算する。これは、c_iの遅延をΔだけ摂動した後の目的関数Ｏの値の変化を測る。次に、類似の感度値を有する通信インスタンスを同じパーティションに割り当てる。この例では、図１１に示される$s(c_i)$の値に基づいて、c_1をCP_1に割り当て、c_2およびc_4をCP_2に割り当て、c_3をCP_3に割り当てる。前述のように、同じパーティション内のイベントは、ＣＡＴによって同様に扱われる。
【００６９】ｂ）プロトコルパラメータの修正
このセクションでは、本発明の方法のステップ３および４について、すなわち、各パーティションを調べる方法、および、その後、最適化されたプロトコルパラメータ値をそれらに割り当てる方法について、説明する。この説明は、各パーティションに割り当てるべき優先度を決定する場合に限定するが、他のプロトコルパラメータ（例えば、バーストモードをサポートすべきかどうか、そしてもしサポートすべきである場合、正しいＤＭＡサイズはどのくらいか）を含むように拡張することが可能である。

【００７０】パーティションの感度は、そのパーティションのイベントがシステムのパフォーマンスに対して及ぼす影響を示す。パーティションの感度のみに基づいて優先度をどのように割り当てたとしても、最適な割当てにはならない可能性がある。その理由は、感度は、１つの通信イベントあるいはイベントのセットが、他の並行する通信イベントの遅延に及ぼす間接的効果（このような効果は、通信アーキテクチャにおける共有チャネル／バスの存在により生じる）を捕捉しないからである。これは、他のパーティション内の通信イベントの遅延に悪影響を与える可能性のあるパーティションにペナルティを課するメトリックを導出することにより考慮される。この情報を得るため、パーティション対（CP_i，CP_j）ごとにＣＡＧを解析し、CP_iに属する通信イベントがCP_jのイベントにより遅延させられる時間を評価＊する。表１に、３個のパーティションを有するシステムに対するデータ例を示す。列２は、各パーティションの感度を与える。列３、４および５は、パーティションCP_i、CP_2およびCP_3内のインスタンスが他の各パーティション内のインスタンスを待機する全時間（w_{ij}）を与える。例えば、CP_1内のインスタンスは、CP_2のインスタンスが終了するための１００サイクルの全待機時間を引き起こす。列６は、パーティションCP_i内のイベントが他のパーティションに引き起こした全待機時間（W_i）を示す、列３、４および５の総和を与える。例えば、CP_2がCP_1およびCP_3に対して引き起こす全待機時間は７サイクルだけである。

【００７１】表１：パーティションの統計量

【表１】

パーティション	感度 $S(c_i)$	w_{i1} クロックサイクル	w_{i2} クロックサイクル	w_{i3} クロックサイクル	w_i クロックサイクル	優先マッピング
CP_1	100	0	100	3	103	17.18=>2
CP_2	85	4	0	3	7	23.57=>1
CP_3	10	0	7	0	7	-75.0=>3

【００７２】これらの統計的パラメータを組み合わせて、最適な優先度割当てを生成する公式とする理想的な方法を求めることは、解くのが困難な最適化問題である。その代わりに、パーティションの優先度を、その感度に比例するように増大させ、一方、他のパーティションに引き起こす待機時間w_{ij}だけペナルティを課すような発見的計算を用いる。表１の記法を用いると、パーティションCP_iの優先度を次のように定義する。

【数１】

$$Pi = V(s(i) - \sum_{j=1}^{n} \frac{s(j)wij}{Wi}$$

【００７３】この公式で、第１項はパーティションCP_iの感度を考慮し、和は、他を妨げることに対してパーティションにペナルティを課す。関数Ｖ（ｘ）は、相対的順序を保持しながら、和の結果ｘを小さい整数P_iにマッピングする（すなわち、$x_i > x_j$ならば、$P_i > P_j$である）。表１の列７は、与えられた値に対するこのマッピングを示す。

【００７４】ｃ）最適な通信プロトコルの合成

このセクションでは、コンポーネントによって生成される通信イベントの各インスタンスをパーティションに分類するハードウェアを合成する方法について説明する。この分類は、コンポーネントの制御フローの短期履歴から導出される、コンポーネントの現在状態に基づく。

【００７５】ＦＳＭ合成の手続きを例で説明する。図１２に、感度に基づくパーティショニングステップを実行した後のＣＡＧの抜粋を示す。簡単のため、通信頂点のみを図中に示す。コンポーネント$Comp_i$の実行トレースにおいて、すべての強調された頂点は、パーティションCP_iに属する。ＣＡＴハードウェア合成ステップにおいて、目標は、頂点がCP_iに属するときに１となりそれ以外のときに０となるブール式を生成することである。複雑さの低いハードウェア実装を得るために、唯一の要件は、このブール式が、選択されたパーティショニング方式をできるだけよく近似することである。図１２における次の３個の波形は、相異なるイベントt_1、t_2、t_4を、コンポーネント$Comp_i$のトレーサとして作用するように選択した３つの場合を示す。トレーサのそれぞれの選択に対して、各通信インスタンスごとに、前のトレーサトークンのインスタンスからの通信インスタンス数によって与えられる距離を計算する。

【００７６】パーティション割当てを実行するため、ＣＡＴハードウェアは、トレーサを検出し、ｘ個の通信インスタンスをカウントして無視し、次のｐ個のインスタンスを特定のパーティションに割り当てることを開始する。例えば、$Formula_i$の場合、t_1がトレーサであり、ｘ＝４であり、ｐ＝３である。図１２に、３つの式（Ｆｏｒｍｕｌａ）のそれぞれから得られる通信インスタンスの実際の分類を示す。図１３に、テストの下でのそれぞれの式（Ｆｏｒｍｕｌａ）の予測の精度を示す。$Formula_1$が最も良好であり、０．９の確率で、与えられたインスタンスがパーティション１（$Partition_1$）に属するか否かを予測する。

【００７７】それぞれの式は、始点としてのトレーサと、通信イベントの発生数に関するカウントとに関連し、したがって、正規表現で表すことができる。その結

果、これは、有限状態機械（ＦＳＭ）としてのハードウェア実装に直接に変換することができる。図１４に、Ｆｏｒｍｕｌａ₁を実装したＦＳＭを示す。

【００７８】一般に、適当なトレーサトークンと、ｘおよびｐに対する適当な値とを選択することは、簡単な仕事ではないことがある。この問題は、回帰理論からの周知の問題によって定式化され、既知の統計技法を用いてそれを解く。

【００７９】それぞれの調べられるトレーサに対して、距離d_1，d_2，...，d_nと、各d_iに対してトレーサトークンから距離d_iにある通信インスタンスがパーティションＣＰ$_i$に属するか否かを示す０または１の値（パーティショニングされたＣＡＧから導出される）とからなるデータセットをＣＡＧから構成する。回帰関数ｆは次のように定義される。

【数２】

$$f(d,\theta)=\begin{cases}1:\theta_1<d<\theta_2\\0:elsewhere\end{cases}$$

【００８０】ｆが１であるとき、距離ｄにあるインスタンスはＣＰ$_i$に属することを示す。二乗誤差

【数３】

$$\sum_{i=1}^{n}|y-f(d,\theta)|^2$$

を最小にする割当て$\theta=\{\theta_1,\theta_2\}$（ただし、$\theta_1=x$、$\theta_2=x+p$）が要求される。ただし、ｙはデータセットからの値であり、ｆ（ｄ，θ）は予測である。回帰関数はθに関して非線形であるため、明示的な解を計算する普遍的な技法は知られていない。しかし、使用可能ないくつかの発見法および反復手続きが存在する（G. A. F. Seber, C. J. Wild, Non-linear Regression, Wiley, New York, 1989、参照）。

【００８１】注意すべき点であるが、一般に、この回帰関数は、コンポーネントからの部分的内部状態や、システムによって処理されるデータのプロパティ（例えば、ＱｏＳスタンプやデッドライン値）のような、設計者が指定する追加パラメータを利用して構成することも可能である。

【００８２】［ＩＶ．Ｄ．実験結果］本発明の技術を、ＴＣＰ／ＩＰネットワークインタフェースカードシステムや、出力キューＡＴＭスイッチのパケット転送ユニットを含む、いくつかのシステム例に適用した結果について説明する。それぞれの例について、システムレベルのコシミュレーションに基づくパフォーマンス結果を提示する。

【００８３】第１の例は、例１に関連して従来の技術のセクションで説明したＴＣＰシステムである。第２の例は、出力キューＡＴＭスイッチのパケット転送ユニット（図１５に示す）である。このシステムは、４個の出力ポートからなり、各出力ポートは、キューに入れられたパケットアドレスを記憶する専用の小さいローカルメモリを有する。到着するパケットビットは、デュアルポート共有メモリに書き込まれる。各パケットの開始アドレスは、スケジューラによって、適当な出力キューに書き込まれる。各ポートは、パケットの存在を検出するために、そのキューをポーリングする。キューが空でない場合、ポートは、デキュー(dequeue)信号をそのローカルメモリに発行し、デュアルポート共有メモリから関連するパケットを取り出し、それをその出力リンクに送る。

【００８４】次の例ＳＹＳは、４コンポーネントシステム（図１６に示す）であり、各コンポーネントは、共有メモリにアクセスするための独立の並行する要求を発行する。図１７に、４個のコンポーネント、２個のメモリ、および、ブリッジにより接続された２本のバスからなるもう１つの例ＢＲＤＧを示す。コンポーネント自体はそれぞれ一方のバスに接続されるが、ブリッジを介してリモートメモリにアクセスするための要求をローカルバスアービタに対してすることができる。また、コンポーネントは、互いに、専用リンクを介して同期する。

【００８５】表２に、静的優先度に基づく通信プロトコルによるＣＡＴベースの通信アーキテクチャを用いる場合のパフォーマンスの利益を示す（On-Chip Bus Development Working Group Specification I Version 1.1.0. VSI Alliance, Aug. 1998、参照）。表の各行は、前述のシステム例のうちの１つを表す。各システムについて、列２は、パフォーマンスメトリックを定義する。ＴＣＰ、ＳＹＳおよびＡＴＭの場合、これらは、システムを通る各データに関連するデッドラインのセットから導出される。それぞれの場合における目的は、これらの例のデッドライン超過数を最小にすることである。ＢＲＤＧの場合、各データトランザクションに重みが割り当てられる。システムのパフォーマンスは、各トランザクションの処理時間の重みつき平均として表される。この場合における目的は、重みつき平均処理時間を最小にすることである。静的通信プロトコルは、各通信要求に対する固定ＤＭＡサイズと、静的優先度に基づくバスアービトレーション方式とからなる。これらの例では、パーティションを識別し優先度およびＤＭＡサイズを割り当てるＣＡＴ方式は、セクション３に記載したようなそれぞれの要求やデッドラインに対する重みのようなユーザが指定する情報を利用して、さらにフレキシブルな通信プロトコルを提供する。

【００８６】表２：ＣＡＴベースのアーキテクチャを用いたシステムのパフォーマンス

【表２】

25　　26

システム例	パフォーマンスメトリック	入力トレース情報	静的プロトコル	ＣＡＴベースアーキテクチャ	パフォーマンス改善
ＴＣＰ／ＩＰ	デッドライン超過	２０パケット	１０	０	－
ＳＹＳ	デッドライン超過	５７３トランザクション	４１３	１７	２４．３
ＡＴＭ	デッドライン超過	１６９パケット	４０	１６	２．５
ＢＲＤＧ	平均実行時間（サイクル）	１００００クロックサイクル	304.72	２５４．１	１．２

【００８７】各システムについて、列４は、静的通信プロトコルを用いて得られるパフォーマンス結果を示し、列５は、ＣＡＴベースのアーキテクチャをシミュレートすることによって生成された結果を示す。高速化を列６に示す。これらの結果によれば、固定パラメータ値を用いたプロトコル上でＣＡＴベースのアーキテクチャを用いることによって、パフォーマンスの大幅な改善が得られる。ＴＣＰ／ＩＰの場合、デッドライン超過数は０に縮小され、一方、ＳＹＳの場合には、２４倍のパフォーマンス改善（デッドライン超過数の縮小に関して）が観測された。

【００８８】効率的なＣＡＴベースの通信アーキテクチャの設計は、図１０のアルゴリズムのさまざまなステップを実行する際の、良好な代表的トレースの選択に依存する。しかし、本発明のアルゴリズムは、通信アーキテクチャを設計するために用いられる入力トレースに固有でない通信アーキテクチャを生成しながら、広範囲の通信トレースにわたりパフォーマンスを改善しようとする＊ものである。ＣＡＴベースの通信アーキテクチャを通じて得られるパフォーマンス改善の入力トレース感度を解析するために、次の追加実験を実行した。ＳＹＳの例について、システムを、大きく異なる特性を有する３つの異なる入力トレースに対して、ＣＡＴベースの通信アーキテクチャおよび従来の通信アーキテクチャでシミュレートした。入力トレースのパラメータは、実行時予測不能性をシミュレートするようにランダムに選択した。すべての場合において、ＣＡＴベースの通信アーキテクチャを有するシステムは、従来の通信アーキテクチャに基づくシステムに比べて、一貫して大幅な改善を示した。これは、ＣＡＴベースのアーキテクチャのパフォーマンスが、入力刺激における変化に対して過度に敏感でないことを示す。その理由は、システムの変化する要求に適応することが可能であるからである。

【００８９】表３：入力における変化に対するＣＡＴベースのアーキテクチャの耐性

【表３】

システム例ＳＹＳへの入力	入力トレース情報	静的プロトコル	ＣＡＴベースアーキテクチャ	パフォーマンス改善
トレース１	８４８トランザクション	３１８	１６１	１．９８
トレース２	５７３トランザクション	４１３	１７	２４．３
トレース３	１０７０トランザクション	３１６	３８	８．３７

【００９０】以上、本発明の実施例について説明したが、上記の記載から他の変形例を考えることは当業者には明らかである。すなわち、上記では本発明のいくつかの実施例についてのみ具体的に説明したが、本発明の技術思想および技術的範囲を離れることなく、さまざまな変形例を考えることができる。

【００９１】

【発明の効果】以上詳細に説明したように、本発明によれば、基礎となる通信アーキテクチャトポロジーを、接続されるコンポーネントの変化する通信要求に適応可能にすることによって、最適化することができる。例えば、重要なデータを別個に処理することにより、通信レイテンシを小さくすることが可能である。この結果、システム全体のパフォーマンス、観測される通信帯域幅およびバス利用率、ならびに、重要なデッドラインを守るシステムの能力などの、さまざまなサービス品質（ＱｏＳ）が大幅に改善される。

【図面の簡単な説明】

【図１】（ａ）は、従来のバス型通信アーキテクチャを用いたＴＣＰシステム仕様の例を示す図である。（ｂ）は、従来のバス型通信アーキテクチャを用いたＴＣＰシステム実装の例を示す図である。

【図２】さまざまなバス優先度割当てに対するＴＣＰシステムの実行を示す図である。

【図３】ＴＣＰ例に対するＣＡＴベースの通信アーキテクチャを示す図である。

【図４】ＴＣＰシステムに対するＣＡＴベースのアーキテクチャの実行を示す図である。

【図５】重要な通信イベントの識別におけるトレードオフを例示するデータ暗号化システムの図である。

【図６】図５のシステムに対するトレースアクティビティの図である。

【図７】分類に用いられるいくつかの変数に関するさまざまな分類メトリックのプロットを示す図である。

【図８】（ａ）は、ＣＡＴベースの通信アーキテクチャを有するシステム例の図である。（ｂ）は、ＣＡＴを用＊いた図８（ａ）のコンポーネントの詳細図である。

【図９】ＣＡＴ最適化された通信アーキテクチャ実行の記号的説明図である。

【図１０】ＣＡＴベースの通信アーキテクチャを設計する手続きの実装を示す図である。

【図１１】ＣＡＣにおける感度計算およびインスタンスの分割を示す図である。

【図１２】代替プレディクタストラテジを示す図である。

【図１３】プレディクタのパフォーマンスを示す図である。

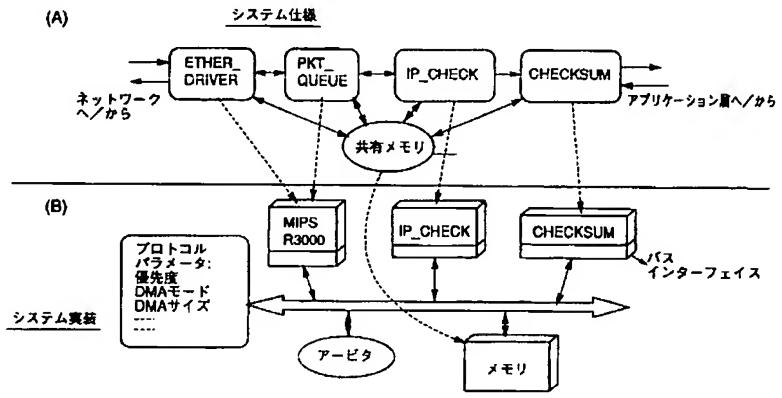
【図１４】Formula$_1$のＦＳＭ実装を示す図である。

【図１５】出力キューＡＴＭスイッチの図である。

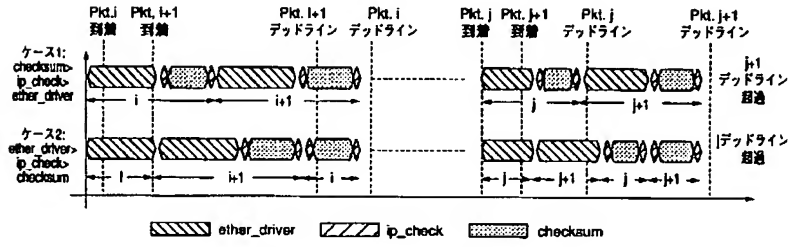
【図１６】並行バスアクセスを有するシステム例ＳＹＳの図である。

【図１７】複数のバスを有するシステム例ＢＲＤＧの図である。

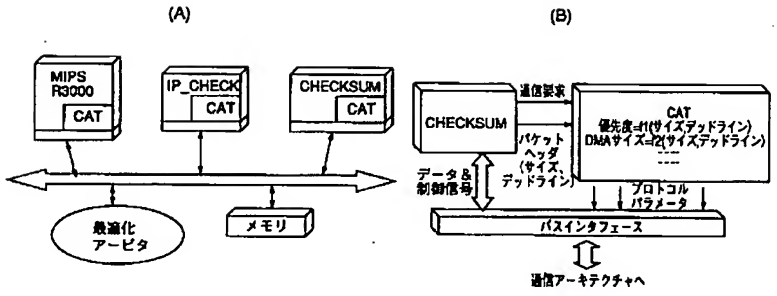
【図１】



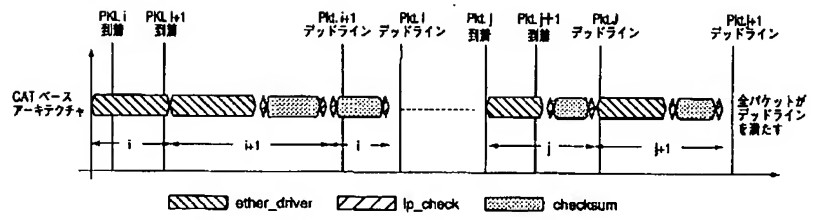
【図２】



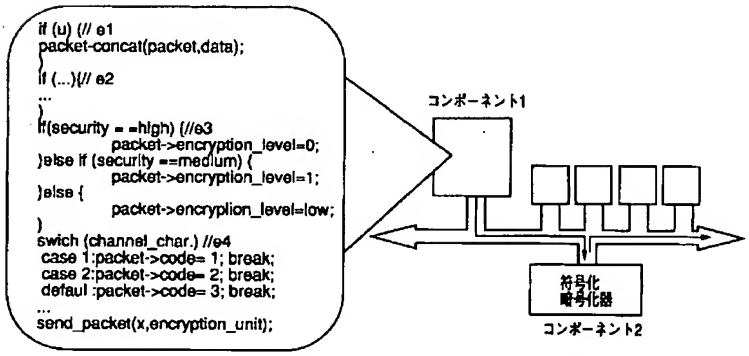
【図３】


(A)
(B)
MIPS R3000
CAT
IP_CHECK
CAT
CHECKSUM
CAT
最適化アービタ
メモリ
CHECKSUM
通信要求
パケットヘッダ（サイズ、デッドライン）
データ＆制御信号
CAT
優先度=f(サイズ,デッドライン)
DMAサイズ=g(サイズ,デッドライン)
プロトコルパラメータ
バスインタフェース
通信アーキテクチャへ

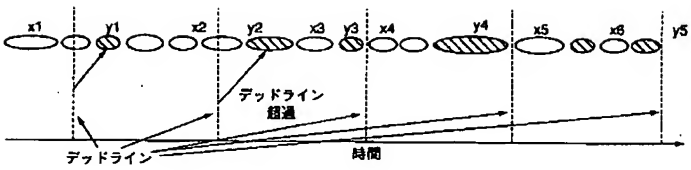
【図４】


Pkt.i 到着
Pkt.i+1 到着
Pkt.i デッドライン
Pkt.i+1 デッドライン
Pkt.j 到着
Pkt.j+1 到着
Pkt.j デッドライン
Pkt.j+1 デッドライン
CATベースアーキテクチャ
全パケットがデッドラインを満たす
i
i+1
j
j+1
ether_driver
ip_check
checksum

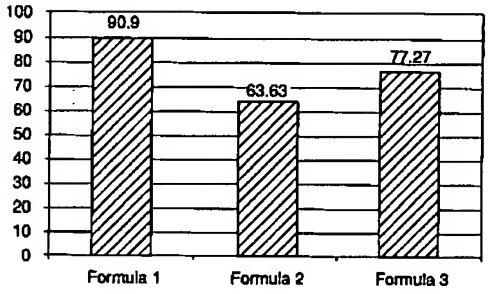
【図５】


if (u) (// e1
packet-concat(packet,data);
}
if (...){// e2
...
}
if(security == high) (//e3
packet->encryption_level=0;
}else if (security ==medium) {
packet->encryption_level=1;
}else {
packet->encryption_level=low;
}
swich (channel_char.) //e4
case 1:packet->code= 1; break;
case 2:packet->code= 2; break;
defaul :packet->code= 3; break;
...
send_packet(x,encryption_unit);
コンポーネント1
符号化
暗号化器
コンポーネント2

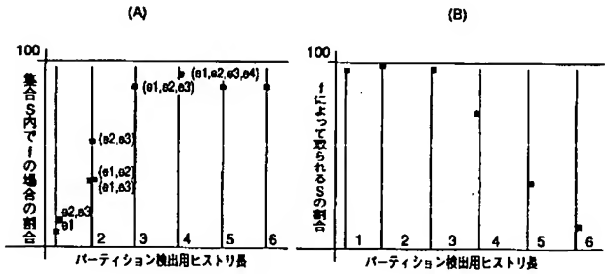
【図６】


x1
y1
x2
y2
x3
y3
x4
y4
x5
x6
y5
デッドライン
超過
デッドライン
時間

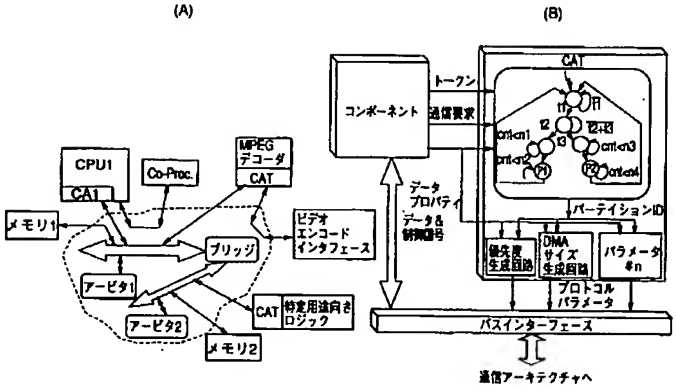
【図１３】


100
90
80
70
60
50
40
30
20
10
0
90.9
63.63
77.27
Formula 1
Formula 2
Formula 3

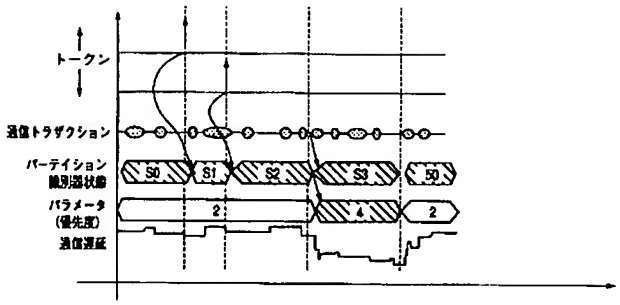
【図７】


(A)
(B)
100
集合S内でiの場合の割合
(e1,e2,e3,e4)
(e1,e2,e3)
(e2,e3)
(e1,e2)
(e1,e3)
e2,e3
e1
1
2
3
4
5
6
パーティション検出用ヒストリ長
100
iによって覆われるSの割合
パーティション検出用ヒストリ長

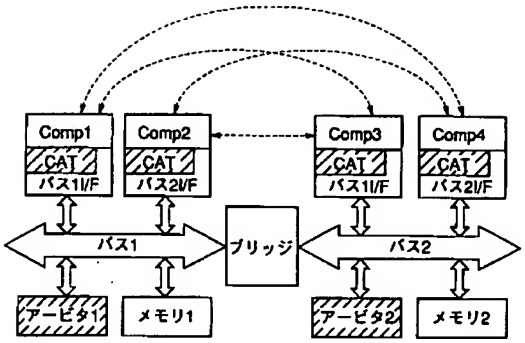
【図８】


(A)
(B)
CPU1
CA1
Co-Proc.
MPEG
デコーダ
CAT
メモリ1
ブリッジ
ビデオ
エンコード
インタフェース
アービタ1
アービタ2
CAT
特定用途向き
ロジック
メモリ2
コンポーネント
トークン
通信要求
データ
プロパティ
データ&
制御信号
CAT
パーティションID
優先度
生成回路
DMA
サイズ
生成回路
パラメータ
#n
プロトコル
パラメータ
バスインターフェース
通信アーキテクチャへ

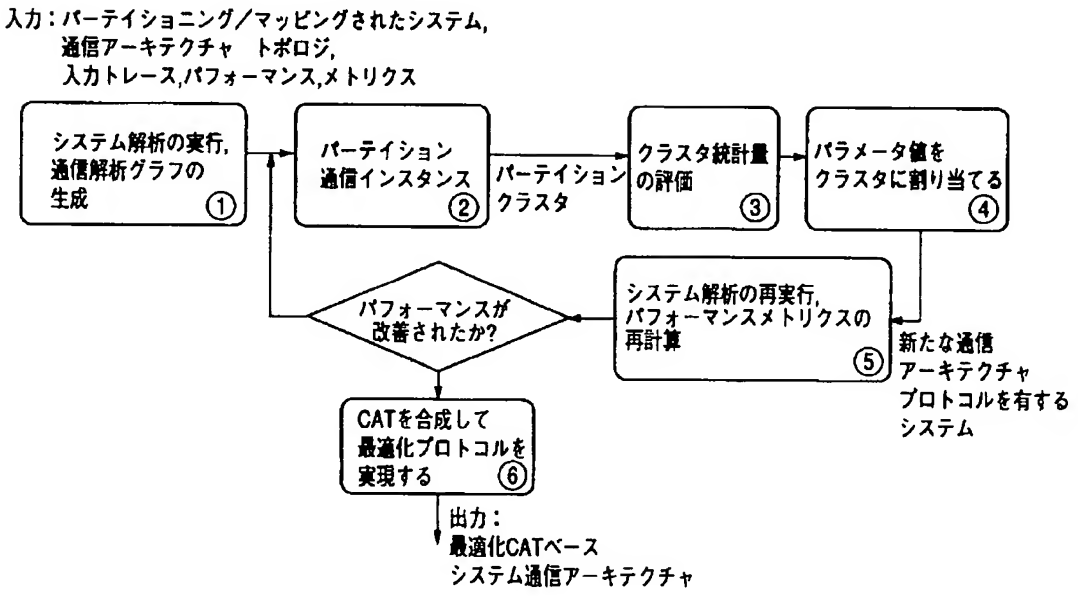
【図９】



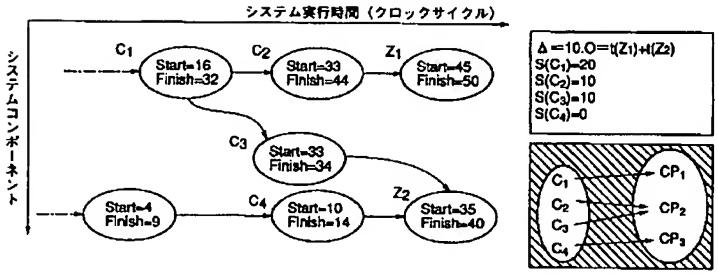
【図１７】



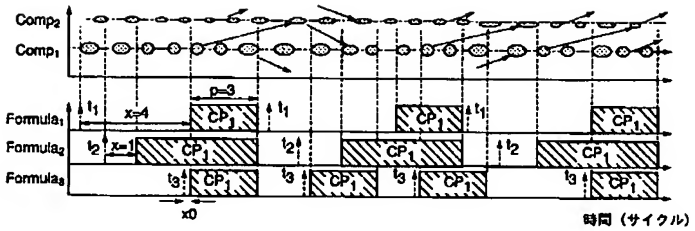
【図１０】



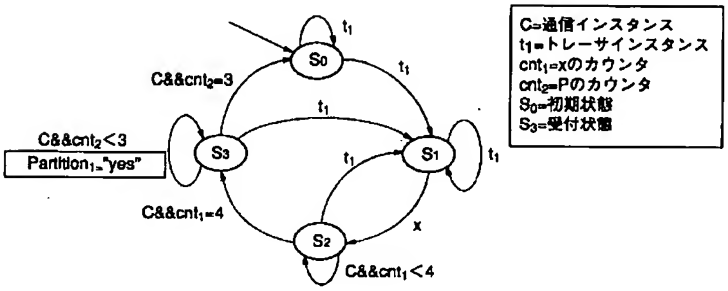
【図１１】



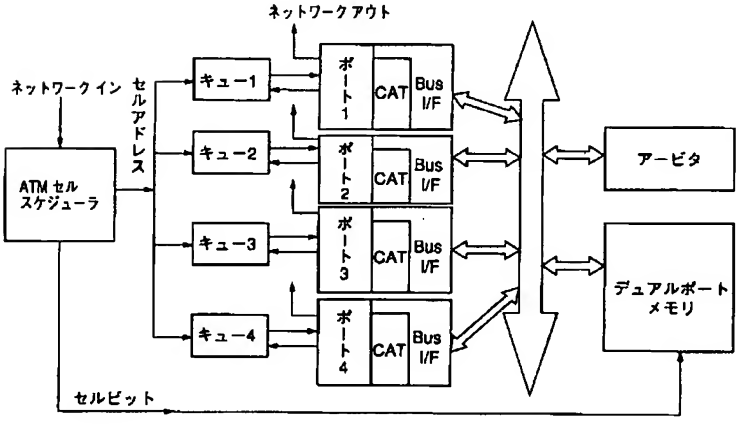
【図１２】


Comp2
Comp1
Formula1
Formula2
Formula3
P=3
x=4
x=1
CP1
t1
t2
t3
x0
時間（サイクル）

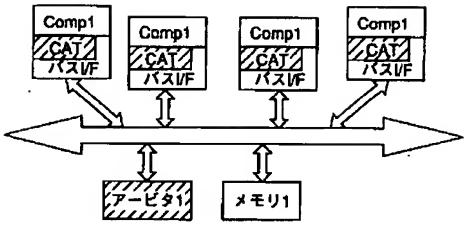
【図１４】


C=通信インスタンス
t1=トレーサインスタンス
cnt1=xのカウンタ
cnt2=Pのカウンタ
S0=初期状態
S3=受付状態
S0
S1
S2
S3
C&&cnt2=3
C&&cnt2<3
Partition1="yes"
C&&cnt1=4
C&&cnt1<4
x

【図１５】


ネットワークアウト
ネットワークイン
セルアドレス
ATM セル スケジューラ
キュー1
キュー2
キュー3
キュー4
ポート1
ポート2
ポート3
ポート4
CAT
Bus I/F
アービタ
デュアルポートメモリ
セルビット

【図１６】



フロントページの続き

(72)発明者　ガネッシュ・ラクシュミナラヤ
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内

(72)発明者　カニシュカ・ラヒリ
アメリカ合衆国，ニュージャージー　08540　プリンストン，４　インディペンデンス　ウエイ，エヌ・イー・シー・ユー・エス・エー・インク内